Web-based Weather Expert System (WES) for Space Shuttle Launch*

T.Rajkumar
SAIC@NASA Ames Research center
MS 269-2, Moffett Field, California, USA
rajkumar@mail.arc.nasa.gov

Jorge E. Bardina
NASA Ames Research center
MS 269-2, Moffett Field, California, USA
Jorge.E.Bardina@nasa.gov

Abstract – *The Web-based Weather Expert System (WES) is a critical module of the Virtual Test Bed development to support "go/no go" decisions for Space Shuttle operations in the Intelligent Launch and Range Operations program of NASA. The weather rules characterize certain aspects of the environment related to the launching or landing site, the time of the day or night, the pad or runway conditions, the mission durations, the runway equipment and landing type. Expert system rules are derived from weather contingency rules, which were developed over several years by NASA. Backward chaining, a goal-directed inference method is adopted, because a particular consequence or goal clause is evaluated first, and then chained backward through the rules. Once a rule is satisfied or true, then that particular rule is fired and the decision is expressed. The expert system is continuously verifying the rules against the past one-hour weather conditions and the decisions are made. The normal procedure of operations requires a formal pre-launch weather briefing held on Launch minus 1 day, which is a specific weather briefing for all areas of Space Shuttle launch operations. In this paper, the Web-based Weather Expert System of the Intelligent Launch and range Operations program is presented.*

Keywords: Weather Tracking, expert system, shuttle launch and range operations.

1 Introduction

The launch weather guidelines involving the Space Shuttle and expendable rockets are similar in many areas, but a distinction is made for the individual characteristics of each [3]. The criteria are broadly conservative and assure avoidance of possibly adverse conditions. For the Space Shuttle, weather forecasts are provided by the U.S. Air Force Range Weather Operations facility at Cape Canaveral beginning at Launch Minus 3 days in coordination with the NOAA National Weather Service Space Flight Meteorology Group (SMG) at the Johnson Space Center in Houston[13, 14]. These include weather trends and their possible effects on launch day. A formal prelaunch weather briefing is held on Launch minus one day which is a specific weather briefing for all areas of Space Shuttle launch operations. Launch weather forecasts, ground operations forecasts and launch weather briefings for the Mission Management Team and the Space Shuttle Launch Director are prepared by the Range Weather Operations Facility. These include all emergency landing forecasts and the end of mission forecasts briefed by SMG to the astronauts, the Flight Director and Mission Management Team [13]. The basic weather launch commit criteria focuses an ambient temperature, wind speed, precipitation[9, 11, 12], lightning [8], type of clouds, and cloud characteristics (cloud temperature and thickness). These factors are used as rules for the weather expert system. There are many rules and conditions depending upon different vehicles. The following sections will explain about the need for an expert system, components of the expert system, and future research directions.

2 Need for a Weather Expert System (WES)

Intelligent Launch and Range Operations Virtual Test Bed (ILRO-VTB) is software for prototyping of large-scale multi-disciplined dynamic systems [1, 4]. With the increasing pressure to integrate and to build control systems and services, there is a desperate need to test and evaluate the complex interactions likely to occur under normal and adverse (i.e. emergency) conditions. The virtual test bed is used to simulate the mission, control, ground-vehicle, launch and range operations. In launch and range operations [1, 4], weather plays a crucial role. The primary goal of the weather expert system is to make expertise available to decision makers who need answers quickly and rapidly in a global manner (macro scale). The expert system [5] can assist with situation assessment and launch planning. The inputs to the expert system can vary across text, numeric and satellite images. The expert system saves money by reducing the time involved in weather analysis and the management can make smarter decisions and work more productively. The complexity and responsibility

*0-7803-7952-7/03/$17.00

of weather operations raises the need for an automated expert system to analyze and provide expertise to the management.

3 Components of Expert System

An expert system [5] is a computer program designed to simulate the problem-solving behavior of a human who is an expert in a narrow domain or discipline. An expert system is normally composed of a knowledge base (information, heuristics, etc.), inference engine (analyzes the knowledge base), and the end user interface (accepting inputs, generating outputs). The path that leads to the development of expert systems is different from that of conventional programming techniques. The concepts for expert system development come from the subject domain of artificial intelligence (AI), and require a departure from conventional computing practices and programming techniques. A conventional program consists of an algorithmic process to reach a specific result. An AI program is made up of a knowledge base and a procedure to infer an answer. Expert systems are capable of delivering quantitative information, much of which has been developed through basic and applied research as well as heuristics to interpret qualitatively derived values, or for use in lieu of quantitative information. Another feature is that expert systems can address imprecise and incomplete data through the assignment of confidence values to inputs and conclusions. One of the most powerful attributes of expert systems is the ability to explain reasoning. Since the system remembers its logical chain of reasoning, a user may ask for an explanation of a recommendation and the system will display the factors it considered in providing a particular recommendation. This attribute enhances user confidence in the recommendation and acceptance of the expert system.

3.1 Knowledge Base

The knowledge the expert uses to solve a problem must be represented in a fashion that can be coded into the computer and then be available for decision making by the expert system. Knowledge bases can be represented by production rules. These rules consist of a condition or premise followed by an action or conclusion (IF condition...THEN action). Production rules permit the relationships that make up the knowledge base to be broken down into manageable units. During the consultation, the rule base is searched for conditions that can be satisfied by facts supplied by the user. This operation is performed by the inference engine. Once all of the conditions (i.e. IF parts of rules) of a rule are matched, the rule is executed and the appropriate conclusion is drawn. Based upon the conclusions drawn and the facts obtained during consultation, the inference mechanism determines which questions will be asked and in what order. There are various inferencing methods available to perform the tasks of searching, matching, and execution. A distinctive characteristic of expert systems that distinguishes them from conventional programs is their ability to utilize incomplete or incorrect data. Given only a partial data set, an expert is likely to have less than absolute certainty in his conclusion. The degree of certainty can be quantified in relative terms and included in the knowledge base. The certainty values are assigned by the expert during the knowledge acquisition phase of developing the system. By incorporating rules in the knowledge base with different certainty values, the system will be able to offer solutions to problems without a complete set of data.

Most of the rules for the weather expert system are derived from weather contingency rules developed over several years by NASA [3]. An example of weather rule is presented here.

if 36 < Temperature < 98 and 0 < Wind Speed < 24 and Precipitaion ="No" and Lightning="No" and Cloud Temperature > 32 and Cloud height > 20000 ft and Cloud thickness < 4500 ft and Cumulus = "No " and Cumulonimbus = "No" then Launch="GO"

The above rule has nine antecedent clauses joined by a conjunction "and" and has a single consequent clause (Launch). The rule is triggered, if all antecedent clauses are set to be true. The clause conditions are derived for each vehicle type. Depending upon the launch vehicle, the rules are slightly changed. The rule variables remain constant for most of the launch vehicles. The rule base consists of rules for GO and NO-GO decisions. Depending upon the prevailing weather conditions, decisions are made.

3.2 Inference Engine

The inference engine looks at the goal variable of the expert system. The inference engine tries to fire that rule by proving the conditions of the rule true using a sequence of rules in the knowledge base. If the system doesn't have enough stored knowledge to prove a rule true, it will ask the end user to supply more information. It continues to search the knowledge base and ask the user for information until a value can be determined for the goal variable. Finally, when it has assembled the knowledge needed to fire a rule that establishes a value for the goal variable, the expert decision is complete. In this paper, the backward chaining mechanism is adopted for the inference engine. The backward chaining is more focussed because it only processes rules that are relevant to the questions. It simply traverses the rule base trying to prove that clauses are true in a systematic manner.

3.3 User Interface

User interfaces are most widely used for interaction with computers because of their ease, superiority, efficiency, user friendliness and robustness as they allow the user to interact by manipulation of graphical objects. In

the weather web expert system, the user interface is automated in such a way that the inputs to the expert system are downloaded and fed to the system in a periodic manner. There is no need of human intervention in the expert system and decisions for launch are automatically displayed as a web page. The weather web expert system is based on Java technology and web enabled, which can be viewed from any part of the world.

4 Weather Expert System

The weather expert system is launched on the web by a dedicated tomcat server[2, 10]. In this section, the details of weather web expert system and acquisition of detailed inputs are discussed.

4.1 User Interface

The user interface is supported by Java servlets [6, 7] where the user can input the data for weather analysis. Java servlet technology provides a simple, consistent mechanism for extending the functionality of a Web server. A servlet can almost be thought of as an applet that runs on the server side. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. In figure 1, there are 16 different buttons. The first four button rows deal on the US and North American continental weather system. The fifth button row provides information on global weather system including tropical cyclones. The "Launch Decision" button activates the expert system and provides the expert decision for the shuttle launch. Except the launch decision button,

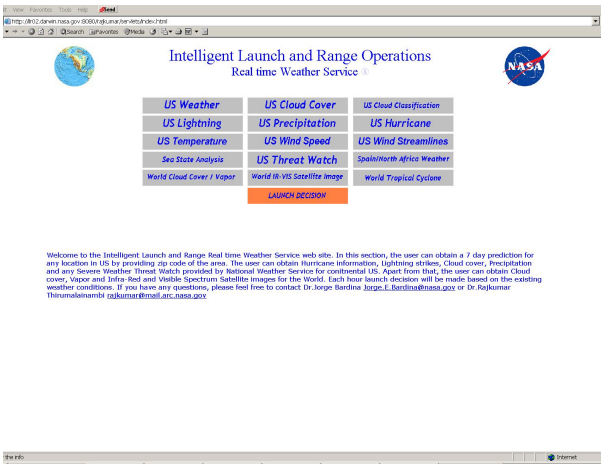

Figure 1: Front-end of Weather Web server

all other buttons invoke corresponding servlets. The servlets get the data or images from various sources across the US. The US weather button provides a 7 day weather forecast for a given zip code in the continental US. It provides a National weather service radar image and GOES satellite image with daily weather forecasting. Apart from these images, specific weather details like humidity, wind speed, barometric pressure, heat index, and dew point are updated in hourly intervals. The US cloud classification is provided by Naval postgraduate school at Monterey, California. The lightning data is provided by National Lightning Detection Network (NLDN) and at 30 minute intervals lightning strikes across the US are updated. The surface temperature contour and surface wind speed (knots) are provided by the National Weather Service. The sea state analysis is provided to the user to understand the booster rocket recovery. Weather criteria for an emergency landing at the TransOceanic Abort Landing Sites (TALS) is monitored by NOAA in Spain and North Africa[13]. The data is downloaded every hour from various agencies in the format of images and numerical data. Once the data is downloaded, the images are processed and specific numerical values are derived for Florida state. The data are used as inputs to the expert system. When a user clicks "Launch Decision button", an expert system inference engine checks the values against the weather rules and it suggests the shuttle launch decision by GO or NO-GO.

In figure 2, the expert decisions for the Shuttle launch are shown below the button group. The green value contributes to the GO situation whereas red value contributes NO-GO situation. If GO should occur, every value in the lower frame should be green. The present

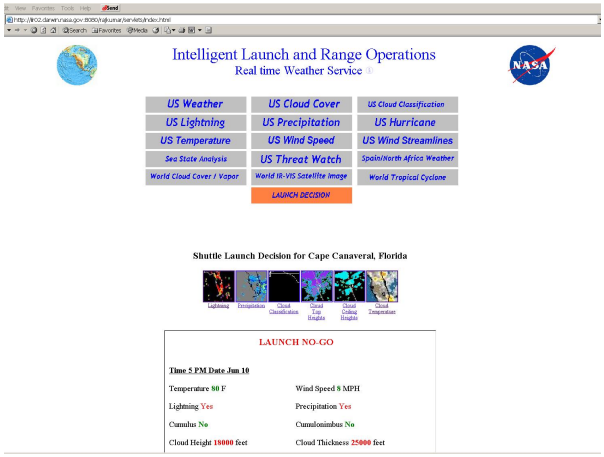

Figure 2: Output of Weather expert system

expert system provides the decision for a generic shuttle launch. For a specific shuttle launch, more stringent rules have to be added to the knowledge base.

4.2 Back End System

The *cron* daemon is a long running process that executes commands at a specific time and date. The real time weather data is obtained from different federal weather monitoring agencies and it is shown in Figure 3.

Figure 4 illustrates the flow of data to the expert system.

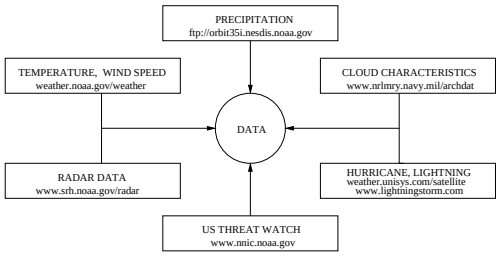

Figure 3: Real-time input data for weather expert system

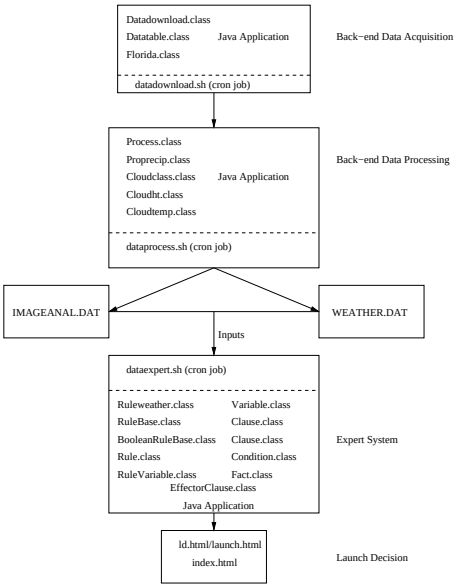

Figure 4: Flow of data for weather expert system

The *datadownload.sh* script executes three Java applications which download images and other types of data. The *dataprocess.sh* script processes the images and extracts and converts to them suitable numerical values. The image processed data are stored in *imageanal.dat* and other numerical values are stored in *weather.dat*. Both data files constitute the inputs for expert system. The *dataexpert.sh* script reads the data and the expert system processes the information and provides a decision for shuttle launch. **Ruleweather** is the main Java class and it acts as an application for the expert system. **Ruleweather** application features a set of controls for selecting backward chain inferencing, the Goal variable and Result when backward chaining is invoked. Only **BooleanRuleBase** supports backward chaining. The **BooleanRuleBase** includes the **Rule** class, the **RuleVariable** class, and the **RuleBase** class as well as supports classes such as **Clause**. The **Rule** class is used to define a single rule and also contains methods that support the inferencing process. Each Rule has a name data member, a reference to the owning **BooleanRuleBase** object, an array of antecedent clauses, and a single consequent clause. The rule's truth value is stored in the Boolean truth. This is a Boolean object, not an elementary boolean variable. The fired boolean member indicates whether this rule has been fired or not. There are several rule constructors, each requiring a reference to the **BooleanRulebase** instance, the Rule name, one or more antecedent or left-hand-side (LHS) clauses, and the single consequent or right-hand-side (RHS) clause. Each constructor allocates the correct number of entries in the antecedents array, and also registers itself with the Clause objects. It also adds the clauses to its data members. The truth is initialized to null, meaning undefined or unknown, and the Rule registers itself with the owning **BooleanRuleBase**. Clauses are used both in the antecedent and consequent parts of the rule. A clause is made up of a **RuleVariable** on the left hand side; a condition, which tests equality, greater than, or less than and the right hand side is a string value ("GO" or "NO-GO"). The **Clause** class contains four methods. The *addRuleRef()* method is used by the **Rule** constructor to register **Rule** with this **Clause**. The *check()* method performs a test of the clause. If the clause is used as a consequent clause, then testing its truth value makes no sense so a null value is returned. If the variable is bound, the switch statement is used to test the specified logical condition and returns the resulting truth value. The *setConsequent()* method sets the consequent **boolean** to true and the *getRule()* method returns a reference to the owning **Rule** instance. The **Condition** class is a helper class to **Clause**. It takes a string representation of a conditional test and converts that into a code for use in the *switch* statement in the *Clause.check()* method.

The **Variable** class has a *name* member to identify the variable, and a **String** value member. The *labels* member is used to hold discrete symbols for categorical variables. The *column* is used to specify the position of the variable in a data file. There is a default constructor, as well as one where the name is specified. Two accessor methods are provided to set the *value* and get the *value* of the **Variable**. The *setLabels()* method is used to define the valid symbolic values for categorical variables.

The *getLabel()* method returns the symbolic value for the specified index and the inverse method *getindex()* returns the index given a symbolic value. The **RuleVariable** provides the support necessary for variables used in inferencing. The constructor takes the *name* of the variable as the only parameter. **RuleVariable** inherits the discrete symbolic behavior of the base **Variable** class. A new data member is the **Vector** *clauseRefs*, which holds references to all **Clauses** that refer to this variable. Instances of **Clause** register themselves by calling the *addClauseRef()* method. There are several methods that are overridden as well as some new ones added for rule processing. The *setValue()* method not only sets the *value* of the variable, it also calls the *updateClauses()* method, which iterates through every **Clause** that refers to this **RuleVariable** and retests its *truth* value via its *check()* method. The *ruleName* holds the name of the rule which set this **RuleVariable's** value. When the rule fires, it calls the *setRuleName()* method. The **BooleanRuleBase** class defines a set of **RuleVariables** and **Rules** along with the high-level methods for backward chaining. The **BooleanRuleBase** has a *name*, a *variableList* that contains all of the **RuleVariables** referenced by the **Rules**, and the *ruleList*, which contains all of the **Rules**. The **BooleanRuleBase** class implements the **RuleBase** interface. It defines a set of methods such as *getGoalVariables()*, *reset()*, and *backwardChain()*.

The **BooleanRuleBase** *backwardChain()* method takes a single parameter, a **String** which is the name of the goal variable. This variable name is used to retrieve the goal's **RuleVariable** instance. All clauses which refer to the goal variable are enumerated and a *while()* loop is used to process each **Clause** object. Rule *backChain()* will make recursive calls to **BooleanRuleBase** *backwardChain()*, if necessary, to follow a chain of inferences through the rule base in order to find out whether the original *goalClause* is true or false. **Rule** *backChain()* will return the **Rule's** truth value. The **Rule** *backChain()* method will try to prove a rule either true or false by recursively calling **BooleanRuleBase** *backwardChain()* until a truth value can be determined. The method consists of a *for()* loop in which each antecedent clause is evaluated in turn. If the variable in an antecedent clause is undefined, then **BooleanRuleBase** *backwardChain()* is called to determine its value. If the clause is true, continue through loop to evaluate the next clause. If it is false, the rule's truth value is false because one of the antecedent clauses is false. If all of the antecedent clauses are true, return true as the **Rule's** truth value. The above classes explain the details of the functionality of expert system, which provides an output as a html file called "ld.html". When the user clicks the launch decision button in figure 1, ld.html is invoked and displays the decision as in figure 2. The value of color indicates the truth value of the rule and corresponding rule is fired.

5 Conclusions

The advantage of the weather expert system (WES) is an unified decision of various weather factors which affects Shuttle launch [2, 5, 9, 10]. The WES acquires data for continental US and World weather. It can be applicable to any launch site in the US by providing the Shuttle launch site rules. The user can define any number of rules and the expert system is very flexible and robust. The WES provides a window of opportunity to launch a Shuttle of a year. The future research focusses on expanding the application of the expert system for various Shuttles and launch sites within the US. Presently the rules are in built in Java program. Future application will provide a rule acquisition system and knowledge building system based on data and humans. The future inference engine mechanism can be based on forward as well as backward chaining technique.

Acknowledgement

The authors wish to thank Nikunj C. Oza and Pramod Gupta, NASA Ames Research Center for their valuable advice and suggestions. Special thanks for the following agencies and institutions for providing data for weather expert system are NOAA, National Weather Service Spaceflight Meteorology group, Naval Research Laboratory, Monterey, California, The Pennsylvania State University, UNISYS and LightningStorm.com.

References

[1] J. Bardina and T. Rajkumar, "Intelligent Launch and Range Operations Virtual Test Bed (ILRO-VTB)", Proc. Aerosense Conference 2003, Orlando, Florida, April 2003.

[2] J.P. Bigus and J. Bigus, *Constructing Intelligent Agents using Java*, John Wiley and Sons Inc. , New York, 2001.

[3] http://chandra.harvard.edu/launch/status

[4] R. D. Davis, *Spaceport Operations Test bed project Description*, Command and Control Technologies Corporation, Florida, 2001.

[5] R. Elaine and K. Knight, *Artificial Intelligence*, McGraw Hill, USA, 1991.

[6] D. Flangan, *Java in a Nutshell*, O'Reilly and Associates, Sebastopol, California, 1996.

[7] J. Hunter, *Java Servlet Programming*, O'Reilly and Associates, Sebastopol, California, 1998.

[8] http://lightningstorm.com

[9] http://science.ksc.nasa.gov/weather/weather.html

[10] M.Watson, *Intelligent Java Applications*, Morgan Kaufmann Publishers, SanFrancisco, 1997.

[11] http://www.met.psu.edu/weather/index_alt.html

[12] http://www.rap.ucar.edu/weather/upper

[13] http://www.srh.noaa.gov/smg/smgwx.htm

[14] https://www.patrick.af.mil